Assignment - 3

# Association Mining - SimplyCast.com

Aastha Suthar, Anshul Hardat, Maurya Shah

13th March, 2020

## Table of Contents

## Executive Summary

This report goes over the process of analyzing user's interactions with the site SimplyCast.com to establish business rules that will help us predict the user's future interactions better to optimize the features and their placements presented to the user. In order to find the association in customer behaviour, we perform analysis on two levels, user and session level data. Looking at the top sets formed from the analysis, we found that the rules and maximally occurring items are quite similar for both user and session data.

## Objective

The objective of this report is to analyze a user's interactions with the site SimplyCast.com to find association in various milestones that the user achieves while interacting with the site. These milestones are achieved at two levels, the user level and the session level. The milestones at user level tell us what are preferences of an average user throughout their account's life time, whereas session per day tells us what most users did on a particular day for a given session. These rules will give us a better understanding of what users mostly do and what they do daily so that we can accordingly serve them for maximum benefit of the business.

## Data Summary

The data on the user's click milestones has been provided in the database "dataset03", under the table "rawdataDec15". The following are the fields of this table,

id: Serially generated id created for each milestone achievement.

user_id: User id for each interaction.

milestone_name: Name of a given interaction milestone.

date: Date when this milestone was achieved.

time: Time when this milestone was achieved.

The dataset contains 665,435 rows in total. There are 3,159 unique users in our data set and a total of 24,713 distinct sessions made by these users.

# Data Cleaning and Transformations

The data set is free of NULL and blank values and hence, we do not need to filter any rows. To understand the dataset in depth, we establish what we will be used for user analysis and what will be used for session analysis. The first thing we consider is that this analysis will be done with Apriori algorithm and hence, we need to have unique keys. Further, we need to provide this data to the system as a single row. For user analysis, we considered user_id as the primary key as it uniquely identifies the users and then, we used distinct to ensure that a milestone would only occur once for a user id. As for sessions, it needed something that would work across the repeated milestones within each of the sessions recorded. Hence, we used a combination of user_id and date to form a session_id. To form the extracted dataset for analysis, we ensure that a given row is a distinct combination of session_id and milestone_name.

## User Analysis

### Abstract

distinct(user_id, milestone_name)

### Query

CREATE TABLE user_data AS (SELECT DISTINCT user_id, milestone_name FROM rawdatadec15)

## Session Analysis

### Abstract

distinct(concat(user_id,date) as session_id, milestone_name)

### Query

CREATE TABLE session_data AS (SELECT DISTINCT CONCAT(user_id,date) as session_id, milestone_name FROM rawdatadec15)

The execution of the aforementioned queries results in the creation of tables user_data and session_data. The data from these tables is extracted into csv files user_data.csv and session_data.csv respectively. This data is further loaded into R and transposed to group milestone names by their unique keys as arules(package for apriori) needs the items that share a key to be in a single row.

# Data Analysis

After loading the data sets and transforming them into forms that are acceptable by apriori algorithm, we use the "arules" package. "arules" is an R package used for apriori algorithm. The analysis part is divided into two,

User Level Analysis and

Session Level Analysis.

## User Level Analysis

### Parameter Tuning

Following the loading of user_data.csv and transposing of its data, we read the data in as a basket transaction. Then, we perform an initial exploration of what the data looks with summary.

```
> summary(tr)
transactions as itemMatrix in sparse format with
 3159 rows (elements/itemsets/transactions) and
 112 columns (items) and a density of 0.1105006

most frequent items:
       ManageTab      ProjPreview SimpleProjCreate        ReportsTab          SendNow          (Other)
            1716             1715             1472              1467             1463            31263

element (itemset/transaction) length distribution:
sizes
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
807 241 147 108 101 103 101  71  76  72  70  78  54  56  58  65  55  65  42  50  55  46  48  31  42  40  24  28  23  27  21  32  33  14  22  24  22  17  21
 40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  64
 12  16  14  12  14  11   8  10  11   5   9  10   5   8   5   5   6   3   1   1   1   1   1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.00    7.00   12.38   19.00   64.00

includes extended item information - examples:
            labels
1       ABSplitProj
2      ABSplitTools
3 AccountSettingsA
```

*Figure 1 : Summary of transaction*

To understand the frequency of milestones better, we plot a histogram to show the top 10 milestones by occurence.

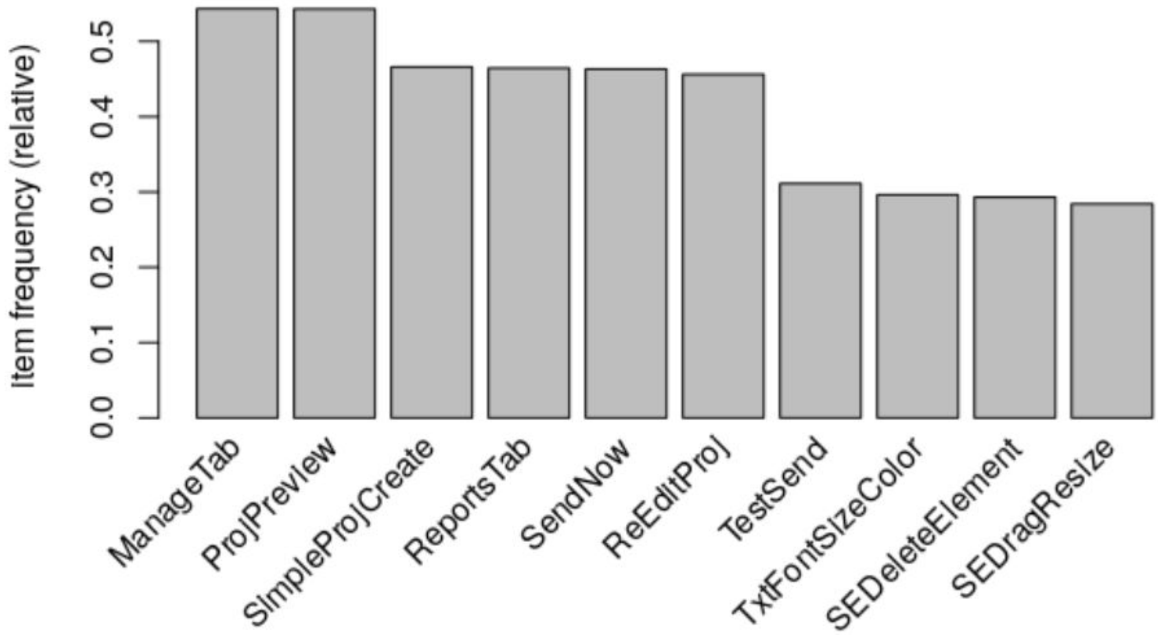


*Figure 2 : Plot of frequent Milestones*

Now that we have grasped what the user data holds, we start using arules. We start with the initial support as 0.4 and confidence as 0.5 as it may provide trustable rules, whilst also providing a considerable number of rules.

```
> rules <- apriori(tr, parameter= list(supp=0.4, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.4      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 1263

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[112 item(s), 3159 transaction(s)] done [0.00s].
sorting and recoding items ... [6 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [8 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 3 : Apriori algorithm try 1*

We get too few rules with this support value(8 rules). Hence, we will reduce our support to 0.3 and test again.

```
> rules <- apriori(tr, parameter= list(supp=0.3, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.3      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 947

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[112 item(s), 3159 transaction(s)] done [0.00s].
sorting and recoding items ... [7 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [76 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 4 : Apriori algorithm try 2*

We get a good number of rules(76 rules). But, we may still get a better number if we go down and hence, reduce the support to 0.2.

```
> rules <- apriori(tr, parameter= list(supp=0.2, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.2      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 631

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[112 item(s), 3159 transaction(s)] done [0.00s].
sorting and recoding items ... [21 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [661 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 5 : Apriori algorithm final (user level)*

The number of rules is now too high(661 rules)! Hence, we roll back to 0.3 and consider it our optimal support value.

## Optimal Rules

The rules are now sorted by their lift and then we inspect the top 15 to get an idea of the formed rules. We do so as higher the lift value, lesser the chance of the rule formed being important because of luck.

```
> inspect(sort(rules,by='lift')[1:15])
     lhs                                    rhs           support   confidence lift     count
[1]  {ManageTab,ReEditProj,ReportsTab}   => {SendNow}    0.3038936 0.9204219 1.987432  960
[2]  {ManageTab,ProjPreview,ReEditProj}  => {SendNow}    0.3206711 0.9200727 1.986678 1013
[3]  {ManageTab,ReEditProj}              => {SendNow}    0.3469452 0.9087894 1.962314 1096
[4]  {ReEditProj,ReportsTab}             => {SendNow}    0.3108579 0.9042357 1.952482  982
[5]  {ManageTab,ProjPreview,ReportsTab}  => {SendNow}    0.3187718 0.8975045 1.937947 1007
[6]  {ManageTab,ReportsTab,SendNow}      => {ReEditProj} 0.3038936 0.8751139 1.918449  960
[7]  {ManageTab,ProjPreview,SendNow}     => {ReEditProj} 0.3206711 0.8747841 1.917726 1013
[8]  {ManageTab,ProjPreview,ReEditProj}  => {ReportsTab} 0.3089585 0.8864668 1.908895  976
[9]  {ManageTab,ProjPreview,ReportsTab}  => {ReEditProj} 0.3089585 0.8698752 1.906964  976
[10] {ReportsTab,SendNow}                => {ReEditProj} 0.3108579 0.8690265 1.905104  982
[11] {ManageTab,ProjPreview}             => {SendNow}    0.3665717 0.8812785 1.902911 1158
[12] {ProjPreview,ReportsTab}            => {SendNow}    0.3266857 0.8805461 1.901329 1032
[13] {ProjPreview,ReportsTab}            => {ReEditProj} 0.3194049 0.8609215 1.887336 1009
[14] {ManageTab,ReEditProj,SendNow}      => {ReportsTab} 0.3038936 0.8759124 1.886167  960
[15] {ManageTab,ProjPreview,SendNow}     => {ReportsTab} 0.3187718 0.8696028 1.872580 1007
```

*Figure 6 : Apriori algorithm result*

## Itemsets Generating Rules

We find all the unique item rule sets that generated our rules to better understand what sets were involved in rule generation.

```
"items" "support"
"1" "{ManageTab}" 0.54320987654321
"2" "{ProjPreview}" 0.542893320671098
"3" "{SendNow,SimpleProjCreate}" 0.322253877809433
"5" "{ReEditProj,SimpleProjCreate}" 0.311490978157645
"7" "{ManageTab,SimpleProjCreate}" 0.330167774612219
"9" "{ProjPreview,SimpleProjCreate}" 0.324786324786325
"11" "{ReportsTab,SendNow}" 0.357708135485913
"13" "{ReEditProj,ReportsTab}" 0.34377967711301
"15" "{ManageTab,ReportsTab}" 0.428300094966762
"17" "{ProjPreview,ReportsTab}" 0.371003482114593
"19" "{ReEditProj,SendNow}" 0.369737258626148
"21" "{ManageTab,SendNow}" 0.41690408357075
"23" "{ProjPreview,SendNow}" 0.397277619499842
"25" "{ManageTab,ReEditProj}" 0.381766381766382
"27" "{ProjPreview,ReEditProj}" 0.397910731244065
"29" "{ManageTab,ProjPreview}" 0.415954415954416
"31" "{ReEditProj,ReportsTab,SendNow}" 0.310857866413422
"34" "{ManageTab,ReportsTab,SendNow}" 0.347261791706236
"37" "{ProjPreview,ReportsTab,SendNow}" 0.326685660018993
"40" "{ManageTab,ReEditProj,ReportsTab}" 0.330167774612219
"43" "{ProjPreview,ReEditProj,ReportsTab}" 0.319404874960431
"46" "{ManageTab,ProjPreview,ReportsTab}" 0.355175688509022
"49" "{ManageTab,ReEditProj,SendNow}" 0.346945235834125
"52" "{ProjPreview,ReEditProj,SendNow}" 0.337448559670782
"55" "{ManageTab,ProjPreview,SendNow}" 0.366571699905033
"58" "{ManageTab,ProjPreview,ReEditProj}" 0.348528015194682
"61" "{ManageTab,ReEditProj,ReportsTab,SendNow}" 0.303893637226971
"65" "{ManageTab,ProjPreview,ReportsTab,SendNow}" 0.318771763216208
"69" "{ManageTab,ProjPreview,ReEditProj,ReportsTab}" 0.308958531180753
"73" "{ManageTab,ProjPreview,ReEditProj,SendNow}" 0.320671098448876
```

*Figure 7 : Sorted result*

### Maximal Frequent Itemsets

Maximal frequent sets are nothing but frequent items where none of it's supersets are frequent. Maximal frequent itemsets help us to make optimal rules by making sure that no further super sets can be formed. This also means that rules that are repeated in the lower subgroups of itemsets formed are not selected and instead of that the biggest superset is considered.
For example, if we have an itemset of A={apple, banana} and then again B={apple, banana, kiwi}, we consider B as maximal set as the set A is a subset of B, and so the rule formed by B covers the rule formed by A.

```
> write(maximal.sets)
"items" "support" "count"
"1" "{TestSend}" 0.311174422285533 983
"2" "{SendNow,SimpleProjCreate}" 0.322253877809433 1018
"3" "{ReEditProj,SimpleProjCreate}" 0.311490978157645 984
"4" "{ManageTab,SimpleProjCreate}" 0.330167774612219 1043
"5" "{ProjPreview,SimpleProjCreate}" 0.324786324786325 1026
"6" "{ManageTab,ReEditProj,ReportsTab,SendNow}" 0.303893637226971 960
"7" "{ManageTab,ProjPreview,ReportsTab,SendNow}" 0.318771763216208 1007
"8" "{ManageTab,ProjPreview,ReEditProj,ReportsTab}" 0.308958531180753 976
"9" "{ManageTab,ProjPreview,ReEditProj,SendNow}" 0.320671098448876 1013
```

*Figure 8 : Maximal Itemsets*

## Session Level Analysis

### Parameter Tuning

Following the loading of session_data.csv and transposing of its data, we read the data in as a basket transaction. Then, we perform an initial exploration of what the data looks with summary.

```
> summary(tr)
transactions as itemMatrix in sparse format with
 24713 rows (elements/itemsets/transactions) and
 112 columns (items) and a density of 0.06253288

most frequent items:
  ManageTab ProjPreview  ReEditProj     SendNow  ReportsTab     (Other)
      13636       11845       11173       11100        8421      116907

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24
3941 1761 1947 2511 2199 1952 1550 1378 1135  981  869  738  570  491  431  353  338  263  241  215  162  144   96  109
  25   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42
  73   67   50   36   17   24   12   17    8    7    8    4    3    4    4    1    2    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   3.000   5.000   7.004  10.000  42.000

includes extended item information - examples:
            labels
1       ABSplitProj
2      ABSplitTools
3 AccountSettingsA
```

*Figure 9 : Summary of data*

To understand the frequency of sessions milestones, we plot a histogram to show the top 10 milestones by occurence.

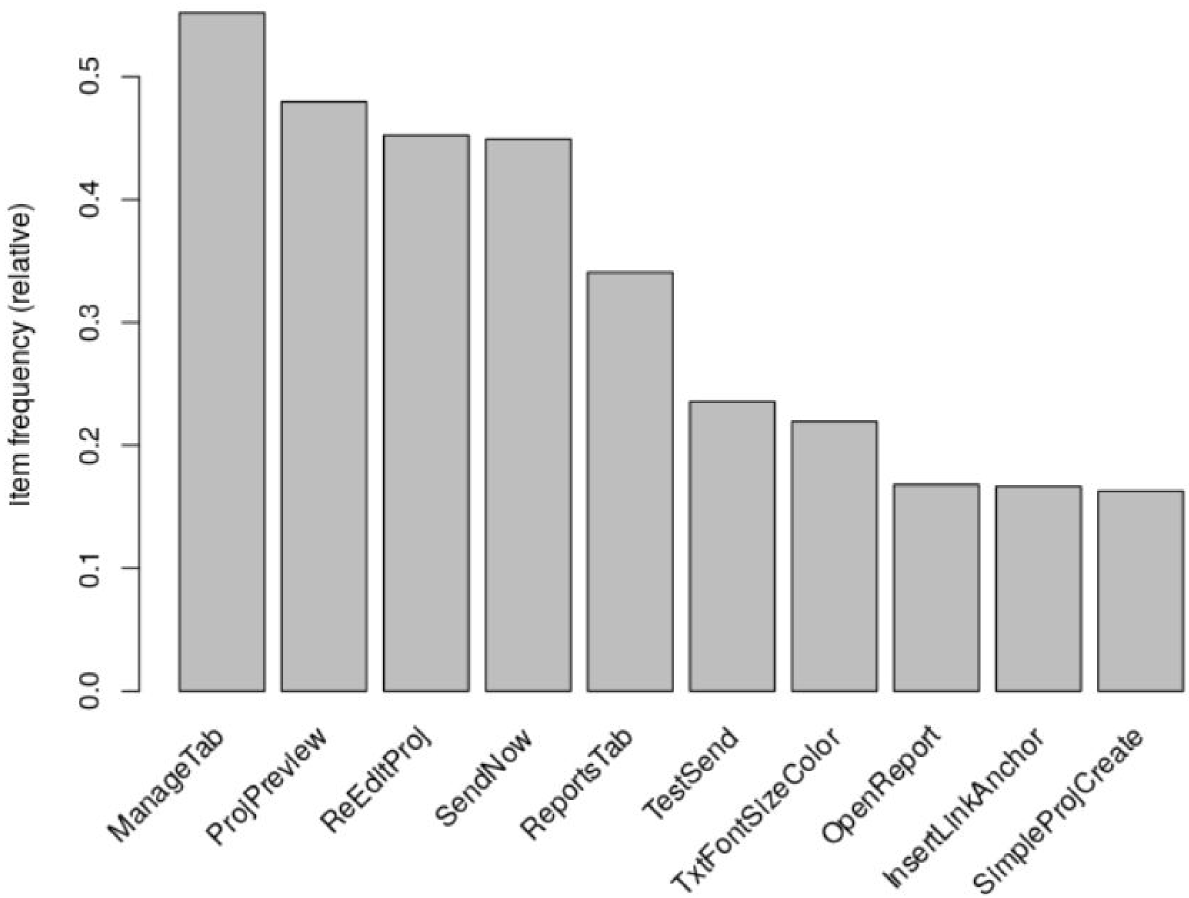


*Figure 10 : Plot of frequent milestone*

Now that we have grasped what the session data holds, we start using arules. We start with the initial support as 0.4 and confidence as 0.5.

```
> rules <- apriori(tr, parameter= list(supp=0.4, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.4      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9885

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[112 item(s), 24713 transaction(s)] done [0.01s].
sorting and recoding items ... [4 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 11 : Apriori algorithm try 1*

We get too few rules with this support value(3 rules). Hence, we will reduce our support to 0.3 and test again.

```
> rules <- apriori(tr, parameter= list(supp=0.3, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.3      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 7413

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[112 item(s), 24713 transaction(s)] done [0.01s].
sorting and recoding items ... [5 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 done [0.00s].
writing ... [5 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 12 : Apriori algorithm try 2*

We still have too few rules with this support value(5 rules). Hence, we dive further down to support value 0.2.

```
> rules <- apriori(tr, parameter= list(supp=0.2, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.2      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 4942

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[112 item(s), 24713 transaction(s)] done [0.01s].
sorting and recoding items ... [7 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [20 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 13 : Apriori algorithm try 3*

The number of rules seems good(20 rules), but we may find more rules if we go down to 0.1, hence, we will do so.

```
> rules <- apriori(tr, parameter= list(supp=0.1, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5     0.1      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 2471

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[112 item(s), 24713 transaction(s)] done [0.01s].
sorting and recoding items ... [23 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [93 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

*Figure 14 : Apriori algorithm try 4*

The number of rules is too high(93 rules), hence, we will consider 0.2 as the optimal support value and use it to generate our rules.

## Optimal Rules

The rules are now sorted by their lift and then we inspect the top 15 to get an idea of the formed rules. We do so as higher the lift value, lesser the chance of the rule formed being important because of luck.

```
> inspect(sort(rules,by='lift')[1:15])
     lhs                          rhs           support   confidence lift     count
[1]  {ManageTab,ReEditProj}    => {SendNow}     0.2413305 0.8135316  1.811244  5964
[2]  {ManageTab,ProjPreview}   => {SendNow}     0.2541982 0.7968037  1.774001  6282
[3]  {ProjPreview,SendNow}     => {ManageTab}   0.2541982 0.9042752  1.638850  6282
[4]  {SendNow}                 => {ManageTab}   0.4047263 0.9010811  1.633061 10002
[5]  {ManageTab}               => {SendNow}     0.4047263 0.7334996  1.633061 10002
[6]  {ReEditProj,SendNow}      => {ManageTab}   0.2413305 0.9000906  1.631266  5964
[7]  {ReEditProj}              => {SendNow}     0.2681180 0.5930368  1.320335  6626
[8]  {SendNow}                 => {ReEditProj}  0.2681180 0.5969369  1.320335  6626
[9]  {ManageTab,SendNow}       => {ReEditProj}  0.2413305 0.5962807  1.318884  5964
[10] {ManageTab,SendNow}       => {ProjPreview} 0.2541982 0.6280744  1.310393  6282
[11] {ProjPreview}             => {SendNow}     0.2811071 0.5864922  1.305764  6947
[12] {SendNow}                 => {ProjPreview} 0.2811071 0.6258559  1.305764  6947
[13] {ProjPreview}             => {ReEditProj}  0.2739854 0.5716336  1.264368  6771
[14] {ReEditProj}              => {ProjPreview} 0.2739854 0.6060145  1.264368  6771
[15] {ReportsTab}              => {ManageTab}   0.2369198 0.6952856  1.260090  5855
```

*Figure 15 : Apriori algorithm resultset*

## Itemsets Generating Rules

We find all the unique item rule sets that generated our rules to better understand what sets were involved in rule generation.

```
> write(itemsets)
"items" "support"
"1" "{ManageTab}" 0.551774369764901
"2" "{ManageTab,ReportsTab}" 0.23691983976045
"3" "{ProjPreview,SendNow}" 0.281107109618419
"5" "{ProjPreview,ReEditProj}" 0.273985351839113
"7" "{ManageTab,ProjPreview}" 0.319022376886659
"9" "{ReEditProj,SendNow}" 0.268117994577753
"11" "{ManageTab,SendNow}" 0.404726257435358
"13" "{ManageTab,ReEditProj}" 0.296645490227815
"15" "{ManageTab,ProjPreview,SendNow}" 0.254198195281835
"18" "{ManageTab,ReEditProj,SendNow}" 0.24133047383968
```

*Figure 16 : Itemsets*

### Maximally Frequent Itemsets

Maximal frequent sets are nothing but frequent items where none of it's supersets are frequent. Maximal frequent itemsets help us to make optimal rules by making sure that no further super sets can be formed. This also means that rules that are repeated in the lower subgroups of itemsets formed are not selected and instead of that the biggest superset is considered.

```
> write(maximal.sets)
"items" "support" "count"
"1" "{TestSend}" 0.235382187512645 5817
"2" "{TxtFontSizeColor}" 0.219236838910695 5418
"3" "{ManageTab,ReportsTab}" 0.23691983976045 5855
"4" "{ProjPreview,ReEditProj}" 0.273985351839113 6771
"5" "{ManageTab,ProjPreview,SendNow}" 0.254198195281835 6282
"6" "{ManageTab,ReEditProj,SendNow}" 0.24133047383968 5964
```

*Figure 17 : Maximal frequent itemsets*

## Conclusion

Thus, using the Apriori algorithm, we have identified associations between the milestones for simplycast, and generated business rules that will eventually help us optimize the arrangement of these milestones. By tweaking the values of support and confidence, it was found that for user level data, the combinations of support of 0.3 and confidence of 0.5 will give us optimal number of rules(76 rules) and the combination support of 0.2 with confidence of 0.5 will give us the optimal number of rules(20 rules) for session. The top rules for either of the analysis gives us a few similar results for the rules like itemset, the item sets generating the rules and maximally occurring items. And lastly, for the maximal itemset, we have 9 rules in user analysis(Figure 8) and 6 rules in session analysis(Figure 17).

# Appendix

## R script

### Main.r

```
library(arules)

library(plyr)

user_data <- read_csv("~/dataset/user_data.csv")

df_user <- user_data

df_user1 = ddply(df_user, c("user_id"), function(dfl)paste(dfl$milestone_name, collapse=","))

df_user1$user_id = NULL

write.table(df_user1, "dataset/user_data_basket.csv", quote=FALSE, row.names = FALSE,
col.names = FALSE)

user_data <- read_csv("dataset/session_data.csv")

df_user <- user_data

df_user1 = ddply(df_user, c("session_id"), function(dfl)paste(dfl$milestone_name, collapse=","))

df_user1$session_id = NULL

write.table(df_user1, "dataset/session_data_basket.csv", quote=FALSE, row.names = FALSE,
col.names = FALSE)
```

### Transformation.r

```
library(arules)

# Reads the csv as baskets

tr <-
read.transactions("~/rstudio/association-mining/dataset/user_data_basket.csv",format="basket",se
p=",")
```

```
# Initial analysis of the data

summary(tr)

# Histogram

itemFrequencyPlot(tr,topN=10)

## User analysis

# To form rules, we need to tune our parameters

# 8 rules

rules <- apriori(tr, parameter= list(supp=0.4, conf=0.5))

# 76 rules - optimal

rules <- apriori(tr, parameter= list(supp=0.3, conf=0.5))

# 661 rules

rules <- apriori(tr, parameter= list(supp=0.2, conf=0.5))

# Setting the optimal number

rules <- apriori(tr, parameter= list(supp=0.3, conf=0.5))

# Shows the rules created

inspect(rules)

# Shows only top 15 rules

inspect(sort(rules,by='lift')[1:15])

# Show item set creating rules

itemsets <- unique(generatingItemsets(rules))

write(itemsets)

# Write all of the rules

write(sort(rules,by='lift'),file="~/rstudio/association-mining/rules/user-rules.txt")
```

```
maximal.sets <- apriori(tr, parameter= list(supp=0.3, conf=0.5, target="maximally frequent itemsets"))

write(maximal.sets)

write(maximal.sets,file="~/rstudio/association-mining/rules/user-maximal-sets.txt")

## Session level analysis

# Reads the csv as baskets

tr=read.transactions("~/rstudio/association-mining/dataset/session_data_basket.csv",format="basket",sep=",")

 # Initial analysis of the data

summary(tr)

# Histogram

itemFrequencyPlot(tr,topN=10)

# To form our rules, we again must tune our parameters

# 3 rules

rules <- apriori(tr, parameter= list(supp=0.4, conf=0.5))

# 5 rules

rules <- apriori(tr, parameter= list(supp=0.3, conf=0.5))

# 20 rules - optimal

rules <- apriori(tr, parameter= list(supp=0.2, conf=0.5))

# 93 rules

rules <- apriori(tr, parameter= list(supp=0.1, conf=0.5))

# Settings the optimal number

rules <- apriori(tr, parameter= list(supp=0.2, conf=0.5))

inspect(sort(rules,by='lift')[1:15])
```

```
# Show itemsets generating rules

itemsets <- unique(generatingItemsets(rules))

write(itemsets)

# Write all rules

write(sort(rules,by='lift'),file="~/rstudio/association-mining/rules/session-rules.txt")

# To get maximally frequent itemsets

maximal.sets <- apriori(tr, parameter= list(supp=0.2, conf=0.5, target="maximally frequent itemsets"))

write(maximal.sets)

write(maximal.sets,file="~/rstudio/association-mining/rules/session-maximal-sets.txt")
```